COMPUTER-AIDED CONTROL ENGINEERING ENVIRONMENT FOR NONLINEAR SYSTEMS ANALYSIS AND DESIGN

James H. Taylor
Control Technology Branch,
General Electric Corporate Research and Development
Schenectady, New York 12345 USA

Abstract. We report on recent progress in developing a computer-aided nonlinear control system analysis and design environment based on sinusoidal-input describing function (SIDF) methods. In particular, two major additions have been made to our CAD software for nonlinear controls during 1984: a simulation-based program for generating amplitude-dependent SIDF input/output models for nonlinear plants, and a frequency-domain nonlinear compensator design package. Both of these are described in detail. This software can treat very general nonlinear systems, with no restrictions as to system order, number of nonlinearities, configuration, or nonlinearity type. An overview of the application of this software to the design of controllers for a realistic, nonlinear model of an industrial robot is presented in Taylor (1984), which serves to illustrate the use of these tools. Based on the software presented here, the use of SIDF-based nonlinear control system analysis and design methods is substantially easier to carry out.

Keywords. Nonlinear control systems, describing functions, control system synthesis, computer-aided design.

1. INTRODUCTION

The basis of this work has been established in earlier publications. In particular, Spang (1982) outlines the basic CAD software tools that were available before the start of this program, Taylor (1982) describes several extensions to that software suite that are required for performing conventional analysis and design for nonlinear systems (equilibrium finding, standard linearization) and required interfaces to create a functionally integrated environment, and Taylor (1983, 1984) establishes the theoretical basis for the work described here.

There are two major additions to our CAD software for nonlinear controls that we developed during 1984: a simulation-based program for generating amplitude-dependent SIDF input/output models for a nonlinear plant, based on constant-amplitude sinusoidal driving signals; and a frequency-domain nonlinear controller design module based on the theory cited above. The specific approach taken in developing this software was:

a. to extend the nonlinear simulation package SIMNON (Elmqvist, 1977), to allow the direct generation of SIDF models (amplitude-dependent frequency response models) using simulation and Fourier analysis methods, as described in Section 2; and

b. to create a Nonlinear Controller Synthesis Program (NCSP) by adding a module to CLADP (Edmunds, 1979) that accepts externally-generated frequency response models (SIDF models or experimental data), and supports nonlinear control system design as described in Section 3.

The basic functional architecture of our environment is portrayed in Fig. 1. The new elements in that figure are the quasilinearizer (SIDF model generator) and the nonlinear design module.

Taylor (1983) provides an overview of SIDF models for nonlinear systems and the overall approach to controller design, and Taylor (1984) outlines and illustrates the methodology for multi-model nonlinear controller design by SIDF inversion. Although the discussion that follows centers on SIMNON and CLADP, the same basic principles can be applied to implement these features in any nonlinear simulation and frequency-domain design package in a straightforward way.

The software developed under this program specifically implements the functions required to carry out two nonlinear control system design techniques from Taylor (1983, 1984), i.e., the *one-model SIDF-based linear controller design method* and the *multi-model nonlinear controller design approach based on SIDF inversion*. These new capabilities provide the basis for dealing with nonlinear systems by adding control system design approaches based on the behavior of the nonlinear system for signals having amplitudes that correspond to the actual anticipated operation. The amplitude dependence of a nonlinear system is a key characteristic that often must be considered in the analysis and design of nonlinear systems. This issue is distinct from the dependence of nonlinear system behavior on operating point, which can often be characterized by a family of standard linearized models about various operating points.

2. A SIMNON-BASED SIDF MODELING PROGRAM

2.1 Functional Overview

Software for generating amplitude-dependent transfer functions describing the input/output (I/O) behavior of nonlinear systems excited by sinusoidal inputs has been developed by extending SIMNON and installing a new built-in system SIDFGEN. The system SIDFGEN provides a sinusoidal driving signal for the nonlinear system, and contains state variables which are Fourier integrals of a selected plant output variable. The SIMNON command set has been extended to support this activity.

The nonlinear plant model can be a single system, or any other arbitrarily interconnected set of subsystems. The only restrictions are that the nonlinear system must be stable, and that the desired SIDF I/O relation must be single-input single-output. If the system is not stable, then the user must stabilize it with the appropriate feedback. If it is desired to obtain a *matrix* SIDF I/O model (e.g., to describe a plant with two inputs and two outputs), then the same principles outlined below can be used, except that the sinusoidal signals

applied to each input must be of different but nearly equal frequencies and the Fourier analylsis must involve integration over an integral number of cycles of both signals. We have successfully determined 2 × 2 SIDF models using input frequencies related by the ratio 4:3; this work will not be described further due to space limitations. Note, however, that one cannot obtain matrix SIDF I/O relations by exciting a nonlinear system one channel at a time, because of the failure of superposition in nonlinear systems; also, exciting both channels with sinusoids of the same frequency is unworkable, because it is impossible to distinguish the effects of multiple inputs on each output.

The input and output of SIDFGEN are designated y_{sig} and u_{cos}, respectively. The signal u_{cos} drives the plant, and the output of the plant is connected to y_{sig}. Once SIDFGEN and the nonlinear plant model are interconnected appropriately, the following excitation is provided by SIDFGEN:

$$u_{cos}(t) = u_0 + a\cos(\omega t) \tag{1}$$

and, by integration of the state variables in SIDFGEN, Fourier integrals for period k = 1, 2, ... are calculated:

$$I_{m,k} = \int_{(k-1)T}^{kT} y_{sig}(t)\exp(-jm\omega t)\,dt; \quad m = 0, 1, 2, 3 \tag{2}$$

where

$$T = 2\pi / \omega \tag{3}$$

The real integral for $m = 0$ captures the constant or "d.c." component of the response, and the complex-valued first-harmonic integral $I_{1,k}$ defines the desired I/O transfer function $G_k(j\omega; u_0, a)$, as follows:

$$y_{0,k}(j\omega; u_0, a) = I_{0,k} / T \tag{4}$$

$$G_k(j\omega; u_0, a) = \omega I_{1,k} / a\pi \tag{5}$$

The remaining higher harmonic Fourier integrals may be calculated to provide the user with a measure of the importance of nonlinear effects and with some insight as to the validity of neglecting higher harmonics. This information is also presented in pseudo-transfer function form,

$$G_{m,k}(jm\omega; u_0, a) = \omega I_{m,k} / a\pi; \quad m = 2, 3. \tag{6}$$

The integrals for $m = 0$, 1 are sampled every period of the sinusoidal input and checked for convergence; when satisfactory convergence is obtained, the simulation is stopped and the Fourier integrals are used to define the I/O relation as in Eqn. (5). Convergence testing is required in this procedure, because transients generally occur in the simulation, and the Fourier integrals are not meaningful until the contributions of these transients have decayed to become small compared to the steady state response. The mechanics of this procedure are discussed in more detail in Section 2.3. Once convergence is achieved, the desired I/O relation is obtained and written to a file to serve as a basis for controller design using NCSP.

2.2 Command Structure

Three new commands have been added to implement SIDF model generation:

a. Frequency selection: The frequencies for which $G(j\omega; u_0, a)$ will be evaluated are determined by the FREQ command. Four variant forms are:

FREQ LIN <ω_{min}> <ω_{max}> <nw> [-ADD] [/<fname2>]

FREQ LOG <ω_{min}> <ω_{max}> <nw> [-ADD] [/<fname2>]

FREQ MAN <ω_1> <ω_2> < ... > [-ADD] [/<fname2>]

FREQ FILE <fname1> [-ADD] [/<fname2>]

where the notation < . > denotes a numeric value and [.] an optional element. The choices are: LIN ("linear"), yielding nw values of ω with equal spacing; LOG, resulting in nw values of ω with equal logarithmic spacing, MAN ("manual"), which accepts the user's list of frequencies directly from the command line, and FILE, which uses the frequency list in an existing file <fname1>.F specified by the user. In all cases, the user may supply a filename <fname2> for storing the frequency list; this file can be used in subsequent SIDF model generation. The option -ADD results in *merging* the newly-specified list with the existing frequency list. If either MAN or FILE is selected, or if the -ADD option is used, then the frequencies are organized into ascending order. The combination of these capabilities allow the user to generate a very finely-tuned frequency list.

b. SIDF model generation: Obtaining the SIDF model is controlled by the FRESP command:

FRESP [<T_{sim}>] [<dt>] [-PART]

which results in evaluating the I/O characterization or frequency response $G(j\omega; u_0, a)$. The optional arguments <T_{sim}> and <dt> are a maximum simulation time and suggested integration step, both of which are discussed in Section 2.3 on convergence. The option -PART results in just the first two integrals being evaluated [Eqn. (2), $m = 0$, 1] and the corresponding first harmonic and dc characterization [Eqns. (4, 5)] being obtained; otherwise all integrals [Eqn. (2), $m = 0$, 1, 2, 3] and I/O relations [Eqns. (4-6)] are evaluated. As the frequency response is being generated, the user is informed of progress by messages of the form FRESP PROCESSING w = <value> as each frequency is considered.

c. Error control parameter setting: The user can control convergence (Section 2.3) by changing the error control parameters using the new command FERR. For example, FERR EPSDC <value> sets ϵ_{dc} to the desired value. Other keywords EPSM, EPSPHI, EPSY, NCYC correspond to the symbols in Section 2.3 in the obvious way.

One SIMNON command has been extended to facilitate the use of this software: DISP, or display. New options are DISP FREQY, to show the list of frequencies defined via FREQ; DISP FREQR, to display the frequency response data; and DISP ECPAR, to show the error control parameters.

Many existing SIMNON commands are useful in SIDF model generation. In particular:

a. Parameter setting: The PAR command may be used to set the parameters a and u_0 in Eqn. (1) to the desired values. The default values for these parameters are 1.0 and 0.0, respectively. If the user inadvertently sets a = 0.0, which is meaningless, then a warning message is issued.

b. Integrability: Several commands in SIMNON can be used to ensure that the nonlinear simulation model can be integrated for sinusoidal excitation of the amplitude and frequencies selected: ALGOR and ERROR. The first selects the integration algorithm, and the second sets the integration error control parameter. Further control can be exercised by the specification of <dt> in the command FRESP, which is a suggested integration step size. The issue of plant model integrability is completely in the hands of the user exercising the existing control mechanisms, as it is in the use of any nonlinear system simulation program. We generally recommend using fourth-order Runge Kutta with self-adjusting step size.

The following example illustrates many features outlined above (everything following " is a comment):

```
SYST NA1 SIDFGEN NACON
"  NA1 is a plant model, NACON connects it
"  to SIDFGEN; set a and u0 appropriately:
PAR A: 1.23
PAR U0: 0.0456
"  Define the frequency list:
FREQ LOG .1 10 7
"  Obtain the SIDF model:
FRESP -PART
"  Display the results:
DISP FREQR
```

results in the display

FREQUENCY RESPONSE, A = 1.23000 U0 = 0.04560

ω	Real G	Imag G	y0
0.1000000	1.029311	-0.1029510	0.01929734
0.2154435	1.011241	-0.2178886	0.01970793
0.4641589	0.9382116	-0.4354827	0.02126725
1.000000	0.7308754	-0.7311587	0.02753362
2.154435	0.4335728	-0.9357857	0.04494397
4.641588	0.2131995	-0.9972967	0.09036508
10.00000	0.0950871	-0.9590450	0.01944327

2.3 Convergence

The convergence testing of the Fourier integrals is a major concern in any attempt to automate I/O transfer function model generation. Since we are dealing with nonlinear simulation, where anything can and often does happen, there is no simple or foolproof answer. Therefore, we will only discuss convergence on a simplified level. Further detail may be found in Taylor (1985).

First, we have had to introduce five new error control parameters in order to provide the user with the required control over convergence. These are ϵ_{dc}, ϵ_m, ϵ_ϕ, ϵ_y, and N_{cyc}. The first three of these are error bounds for the dc component, magnitude, and phase, respectively; ϵ_y defines "small" output signal amplitude in the context of the system being studied; and N_{cyc} provides one mechanism to limit the number of cycles of the input sinusoid that will be used. The first two parameters are relative error bounds and thus must be between 0.0 and 1.0; ϵ_ϕ and ϵ_y are absolute parameters and must be in the ranges $0. < \epsilon_\phi < 30.$ degrees, $0. < \epsilon_y$; N_{cyc} must be greater than two.

Somewhat loosely speaking, the Fourier integrals are said to have converged when the d.c. component $y_{0,k}$ and the magnitude and phase of the transfer function G_k, denoted M_k and Φ_k respectively, have converged in the sense that $y_{0,k}$, M_k, and Φ_k evaluated over the previous ($(K-1)$st) period of the simulation and the latest (Kth) period satisfy the following conditions:

$$err_{dc} = \frac{|y_{0,K} - y_{0,K-1}|}{|y_{0,K}|} < \epsilon_{dc}$$

$$err_m = \frac{|M_k - M_{K-1}|}{|M_k|} < \epsilon_m \qquad (7)$$

$$err_\phi = |\Phi_k - \Phi_{K-1}| < \epsilon_\phi$$

where the default values for the error bounds are $\epsilon_{dc} = \epsilon_m = 0.05$ (these are relative error bounds of 5%) and $\epsilon_\phi = 2$ degrees.

In general terms, there are several things that can prevent convergence. Most often, more simulation time is required; one must inspect y_{sig} to determine if that is so; and if so, increase either N_{cyc} or T_{sim}. The first approach changes the number of cycles that will be used before the convergence algorithm gives up (the default is $N_{cyc} = 5$), while the second changes the maximum simulation time (see FRESP above; the default is an arbitrary value of 10 (seconds or whatever unit of time is used)). Since the simulation will stop after N_{cyc} periods or T_{sim} seconds, *whichever is longer*, it is clear that N_{cyc} governs the run at low frequencies, while T_{sim} limits the simulation time at high frequencies. These two simulation control parameters are provided to help ensure that FRESP does not get "stuck" endlessly integrating a difficult system, to minimize the possibility that the user will be compelled to CTRL-Y the program to regain control; therefore, we recommend that these parameters be set carefully.

Another problem is that there may be a very slow mode which keeps the dc component from converging. If the user is not concerned with the dc component (often it is not important), then convergence can be obtained without an increase in simulation time by increasing ϵ_{dc}.

The next most common convergence problem is that the output sinusoidal component of y_{sig} is "small" in one of the following senses and should be checked:

$$|y_{sig}| < \epsilon_y, \; k = K-1, \; K$$

or (8)

$$M_k < MG_{\max}\epsilon_y, \; k = K-1, \; K$$

where $MG_{\max}$ denotes the maximum value of the magnitude of $G(j\omega; u_0, a)$ for the previous frequencies processed. The first test is used only for the first frequency considered; the second test reflects a better convergence test based on the prior accepted values of $|G|$ rather than on the arbitrary absolute error parameter ϵ_y alone. In both cases, $G(j\omega; u_0, a)$ is declared to be *undefined* (nothing is written to the frequency response file). In dealing with these situations, the first consideration should be ϵ_y, since this is an arbitrary absolute parameter for the first frequency processed by FRESP. The default value (1.E-6) may not have any meaning; if so, then a change is in order using the FERR command. On the other hand, there may be *no* periodic signal component in y_{sig}. This is especially likely if there are biases and saturations in the model. Then it must determined why this is so and whether or not the frequency response has any meaning under those circumstances.

In conclusion, we reiterate that making this procedure completely foolproof is difficult. In particular, additional care must be taken to handle the dc convergence error correctly when y_0 or G_k are zero or very small in some sense.

3. NONLINEAR CONTROLLER SYNTHESIS PROGRAM (NCSP)

The objective was to create a new module NCSP that interfaces with CLADP (Edmunds, 1979) and supports nonlinear control system design. The discussion that follows does not assume a detailed familiarity with CLADP; the reader is referred to Edmunds (1979) for further information. The basic functions of NCSP are:

a. to accept and display a new system description type (frequency response data);

b. to generate a curve-fitted linear analytic model (in transfer function (ratio-of-polynomial) form, based on minimum mean square error curve fitting (Lin, 1982)) corresponding to frequency response data (Section 3.1);

c. to permit the use of existing CLADP capabilities for designing linear controllers in the frequency domain; and

d. to provide a nonlinear controller synthesis capability based on a pre-existing linear controller design (obtained using the capability c. above or otherwise), a corresponding set of frequency response models generated by SIDFGEN (Section 2) for the controller in cascade with the nonlinear plant for various controller input amplitudes, and automatic nonlinear gain synthesis via SIDF inversion [Section 3.2; Taylor (1984)].

This approach to nonlinear design is best suited for dealing with situations where the amplitude-dependence of the nonlinear plant model is of primary concern, rather than situations where different operating points lead to different linearized models. The NCSD routines we have implemented currently handle only zero operating points; the extension to arbitrary values is obvious but not simple to implement.

Like most frequency-domain design programs, CLADP accepts linear models in either Laplace (ratio-of-polynomials) form or state-space (A, B, C, D) form. We extended the model definitions to include direct frequency response models of the sort generated by FRESP (or taken directly from laboratory measurements). This simply involves reading the frequency response data from a file and storing it for further use.

3.1 System Model Conversion by Curve Fitting

Linear compensators based on the plant frequency response model may be directly designed using the classical frequency domain approaches without further model manipulation, or the user may employ any other linear design approach by using the FIT routine to generate ratio-of-polynomial models that best approximate the FR data in the minimum mean square error sense. Since the original data is in the frequency domain, it is most direct to convert to Laplace form; if the user desires to obtain a state-space model, then the appropriate conversion can be made.

The curve-fitting routine we implemented is based on the algorithm given in Lin (1982). Our tests have shown that this approach works well, as long as there is not a large range of ω in which the slope of the magnitude is not an integral multiple of 20 dB/decade. [This phenomenon can be caused by nonlinear effects, see Taylor (1984); it is particularly troublesome in the context of curve fitting if it occurs at low frequencies.] In such cases, the user can control this problem by the appropriate restriction of the frequency range.

In our implementation, the details of the fitting procedure can be specified in advance by using the following commands: NUM <value> (numerator order), DEN <value> (denominator order), BIAS (toggle bias removal switch on/off), FIXN (toggle switch to fix the numerator s^n coefficient to be unity) and FREQ (define frequency range for fitting). Ordinarily, the FIT routine provides a model in which the denominator s^0 coefficient is unity; however, that must be changed if the user has a type-one plant model (one having a pole at $s=0$). If the FIT command is issued directly, the user is asked to supply this information.

The user may find it necessary to iterate in obtaining a meaningful fit to the frequency response data. For example, it may be necessary to vary numerator and denominator order (NUM, DEN), to try restricting the range of frequencies used by the fit algorithm (FREQ), etc., until an adequate fit is obtained.

The command LCD has been installed to provide access to the CLADP linear controller design approaches. The use of this capability may result in a linear controller design that provides satisfactory performance, in which case the user has successfully carried out the one-model SIDF-based linear controller design method (Taylor, 1983). If this is not the case, then the systematic approach outlined in that paper moves on to *multi-model nonlinear* controller design by SIDF inversion. We have implemented one of these approaches, as follows:

3.2 Direct Synthesis of Nonlinear Controllers

The basic idea (Taylor, 1984) is to process a set of frequency-domain models for the nonlinear plant in cascade with a linear compensator designed as above and synthesize a *static nonlinearity* so that the forward path of the resulting control system (see Fig. 2) will be as insensitive to input (error signal) amplitude e as possible, where the set of values $\{e_k, k = 1, 2, ...\}$ is selected by the user, based on the study of the performance of the linear controller that lead to the decision to seek a nonlinear controller and on anticipated operating conditions.

This procedure involves six steps:

a. select one plant operating regime defined by input amplitude a_0 and generate the $G_0(j\omega; a_0)$ model using SIDFGEN, Section 2;

b. design a linear controller of any type denoted $C(j\omega)$ (use CLADP - the primary concern is to achieve good speed of response or bandwidth and transient response); create a SIMNON model of $C(j\omega)$;

c. combine $C(j\omega)$ with the nonlinear plant model and SIDFGEN, drive it with sinusoidal signals of d.c. value e_0 and sinusoidal amplitude e_k, $k = 1, 2, ...$ and use SIDFGEN to generate the transfer function models denoted $\{C(j\omega)\ G_k(j\omega; e_0, e_k)\}$ or, more simply, $\{CG_k\}$;

d. take the set of models $\{CG_k\}$ and for each k determine the precompensating static gain K_k required to force $K_k CG_k$ to just avoid a specified M-circle, as illustrated in Fig. 3;

e. pass the set of gains $\{K_k(e_k)\}$ to the SIDF inversion routine for nonlinearity synthesis; and

f. validate the nonlinear controller design via simulation.

This approach thus results in designing a *general* compensator [in terms of its dynamic response $C(j\omega)$] with *one* nonlinearity that is automatically synthesized. The nonlinearity to be placed in series with $C(j\omega)$ (in front of $C(j\omega)$ in Fig. 2) is synthesized using a number of existing tools; the new elements are items d. (the M-Circle Algorithm) and e. (the SIDF Inversion Algorithm). The latter routines are described below.

3.2.1 Quasilinear gain evaluation (the M-Circle Algorithm). This technique proceeds as follows: The input for this procedure is frequency response data for CG_k, in k files. The segments that define each curve are processed to determine the gain K_k so that $K_k CG_k$ just touches the M circle, as shown in Fig. 3. The details are omitted for the sake of brevity; note, however, that if the resulting K_k is infinite, the user is so informed and a recommendation that a smaller value of M be tried will be issued if CG_k goes into the half plane $U < 0$ at all. (If CG_k is always in the right-half-plane (is positive real), then the "gain margin" is infinite, which is surely conservative; generally there is no problem that would necessitate synthesizing a nonlinear controller is such is the case.) As mentioned previously, the output of this procedure is a set of error amplitude-gain pairs $\{e_k, K_k\}$ from which a precompensating nonlinearity is synthesized.

3.2.2 Controller Nonlinearity Synthesis by SIDF Inversion. For each value of e_k there exists a linear compensator design that differs only in the gain values K_k. This information serves as the basis for nonlinear controller synthesis, as follows: Given the gain versus amplitude relation of the form $K_k(e_k)$ that is to be achieved by a single nonlinearity, adjust the parameters of a specified piecewise-linear nonlinear function $f_{pwl}(e)$ so that the SIDF of f_{pwl} provides the best fit to the gain/amplitude relation $K_k(e_k)$ in the minimum-mean-square-error sense. We defined f_{pwl} in fairly general terms as shown in Fig. 4. The parameters to be adjusted are the slopes, S_1, S_2, the breakpoint, δ, and the step discontinuity, D.

The user is allowed to restrict the nonlinearity by fixing any of the parameters of f_{pwl}; only free parameters will be adjusted for mean square error minimization. For example, fixing

$$S_1 = 0.0,\ S_2 = 0.0$$

restricts f_{pwl} to being a relay with deadzone, of arbitrary output level D and deadzone width 2δ. The parameters shown in Fig. 4 allow enough degrees of freedom that most reasonable gain/amplitude relations can be fit with decent accuracy; allowing the designer to restrict this freedom should permit the user to arrive at a practical design.

In summary, denoting the parameters that define $f_{pwl}(e)$ by the vector p,

$$p^T = [S_1, S_2, \delta, D]$$

we developed a routine for evaluating the SIDF of $f_{pwl}(e,p)$, and the parameter adjustment is done using MINPACK (More, 1980). The output of MINPACK is the parameter set p^* for the nonlinearity in the controller, which completes its definition. It should be observed that the gain set $\{K_k(e_k)\}$ must be well-conditioned, in the sense that the values of e_k must cover a reasonable range of e (e.g. $\{e_k\}$ = ({1.2, 3.3, 5.4, 7.5} rather than $\{e_k\}$ = {1.2, 1.35, 1.50, 1.65}); the reason for this is that the SIDF of the nonlinearity f_{pwl} cannot change abruptly with small amplitude changes, so a closely-spaced gain set will usually be meaningless. Also, the user should be aware that quite different nonlinearities may have very similar SIDFs [cf. Gelb (1968), Figs. B.1 and B.2, where the SIDFs for a 3-level quantizer is almost identical to the SIDF of a limiter for all normalized input amplitudes greater than 1.5]. Therefore, the user may have considerable latitude in choosing the nonlinearity to implement.

A second output is provided so that the nonlinearity can be incorporated directly in a simulation of the closed loop system. This is a SIMNON text file <fname>.T, where <fname> is selected by the user, that models the controller nonlinearity; the parameter values are written according to the results obtained by the nonlinearity synthesis procedure. The user is then ready to simulate the plant plus nonlinear controller in either open or closed-loop configuration by supplying the appropriate connecting system.

For further illustration, see Taylor (1984).

4. SUMMARY AND CONCLUSIONS

The primary motivation for the effort described above is to simplify the use of SIDF-based methods for analysis and design of nonlinear systems, and to transition this technology to control engineers in other parts of General Electric. It is the author's hope that this description will allow others to achieve the same objectives.

ACKNOWLEDGEMENT

Kevin Strobel and Ralph Quan, both employed at GE CRD and students at the Rensselaer Polytechnic Institute in 1983 and 1984, contributed a great deal to the success of this effort.

REFERENCES

Edmunds, J. M. (1979). Cambridge linear analysis and design program. *IFAC Symposium on Computer Aided Design of Control Systems*, Zurich.

Elmqvist, H. (1977). SIMNON - An interactive simulation program for non-linear Systems. *Proceedings of Simulation '77*, Montreux.

Lin, P. L., and Y. C. Wu. (1982). Identification of multi-input multi-output linear systems from frequency response data. *Trans. ASME, 104*, March, 1982.

More, J. J., B. S. Garbow, and K. E. Hillstrom. (1980). User Guide for MINPACK-1, Argonne National Laboratory, Report No. ANL 80-74.

Spang, H. A. III. (1982). The federated computer-aided control design system. *Proc. 2nd IFAC Symp. on CAD of Multivariable Technological Systems*, West Lafayette, IN, 121-129.

Taylor, J. H. (1982). Environment and methods for computer-aided control system design for nonlinear plants. *Proc. 2nd IFAC Symp. on CAD of Multivariable Technological Systems*, West Lafayette, IN, 361-367.

Taylor, J. H. (1983). A systematic nonlinear controller design approach based on quasilinear system models. *Proceedings of the American Control Conference*, San Francisco, CA, 141-145.

Taylor, J. H., and K. L. Strobel. (1984). Applications of a nonlinear controller design approach based on quasilinear system models. *Proc. American Control Conference*, San Diego, CA, 817-824.

Taylor, J. H. (1985). Software tools for nonlinear controller design. GE Corporate Research and Development Report No. 85CRDnnn.

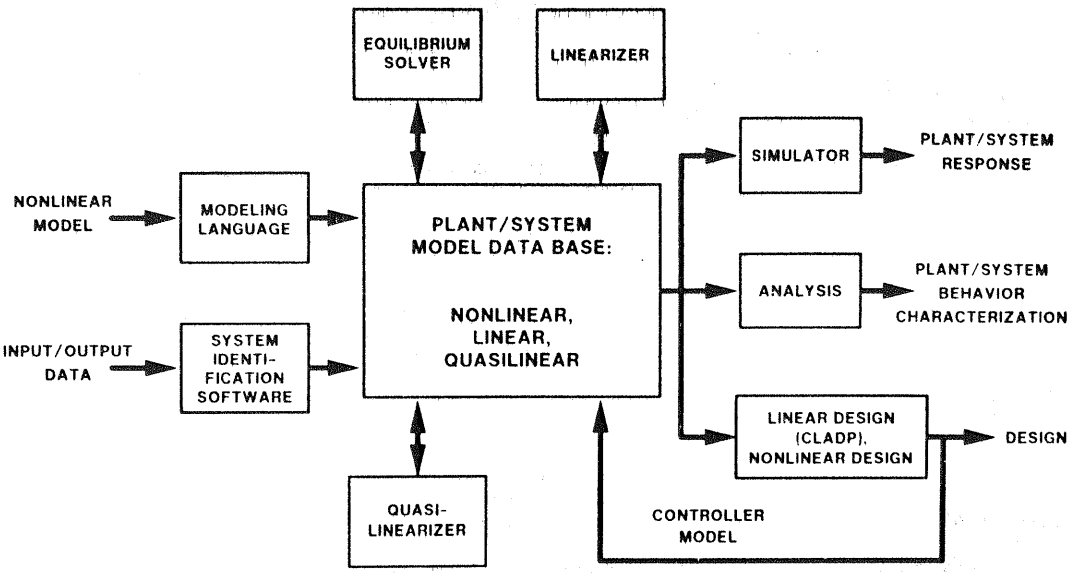

Fig. 1. CAD Environment for Nonlinear System Design.

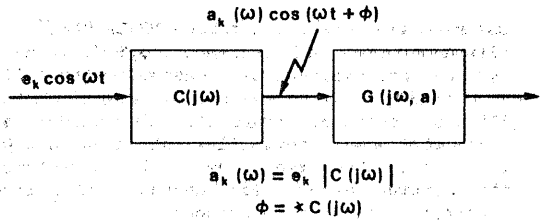

Fig. 2. Compensated Forward Path

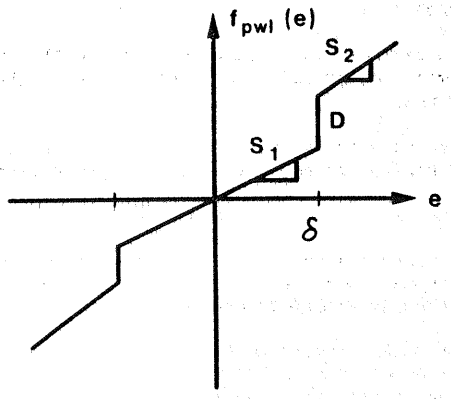

Fig. 4. Piecewise-Linear Compensator Nonlinearity

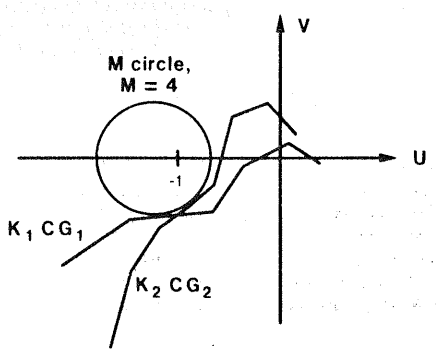

Fig. 3. Illustration of $K_A(c_k)$ Generation